在行動環境中解決範圍查詢之協同快取分享方法

Using Cooperative Cache Sharing Approaches to Answer Range Queries in Mobile Environment

林朝興

南台科技大學資管所

mikelin@mail.stut.edu.tw

張耀華

南台科技大學資管所

m9090218@webmail.stut.edu.tw

黃品菁

南台科技大學資管所

m9390220@webmail.stut.edu.tw

摘要

當行動節點(MH)發出和地理位置有關的範圍查詢(RQ)請求時，基地台(BS)必須追蹤 MH 當時的座標位置，才能求出正確的答案。然而 BS 服務大量的 MH，BS 有限的系統資源將因大量 RQ 計算而被逐漸消耗，造成整體服務效能降低。如今，新的無線通訊技術：MANET 讓 MH 不需要透過 BS 即可以互相分享資料。本文目的為利用 MANET 機制，提出行動節點協同快取分享方法(MCC)，讓 MH 間彼此合作，分享其先前快取過的 RQ 結果，減少 BS 所要處理的 RQ 數或者縮小 RQ 範圍，降低 BS 重覆計算的負擔，提升 BS 整體服務效能。另外，本文亦根據快取資料的利用率、存取率及更新率，改善 MCC 快取置換的效能。模擬實驗結果證實透過 MCC 可以明顯的減少 MH 向 BS 發出的 RQ 數，降低 BS 負擔，提升 BS 整體效能。

關鍵詞：行動計算、範圍查詢、快取合作

Abstract

When numerous mobile hosts (MHs) issue range queries to the base stations (BSs) at the same time, the BSs must consume numerous resources to calculate those queries and that will abate the performances of the BSs. If these query results can be reused, the BSs will avoid calculating these queries repeatedly. In this paper, we propose a Mobile-host Cooperative Caching (MCC) to improve the service performances of the BSs. MHs can avoid issuing new queries or can narrow the query range by utilizing the results of nearby MHs. Consequently, BSs can reduce the loads for calculating these queries and that will improve the service performance. We also propose a cache replacement approach to improve the cache performance of MCC. The experiment results show that the MCC can effectively reduce the query requests and the loads of the BS.

Keywords: Mobile Computing, Range Query, Cooperative Caching

1. 緒論

無線通訊科技技術的快速成長及基地台的廣泛架設，使得人們可以隨時隨地透過行動通訊裝置，例如手機、個人數位助理、筆記型電腦…

等，傳送以及接收資訊。近年來，行動通訊裝置其行動計算能力、儲存空間大幅提升以及多功能的用途，再加上價格趨近於平價，為一般大眾能力所及，於是行動通訊裝置已經普及成為個人隨身的配備。因此許多相關的個人通訊服務(PCS)應用也競相推出，包括了全球定位系統(GPS)、地理資訊系統(GIS)、行動商務(MC)…等[1,13,14,11]。

在 PCS 環境中，基地台(Base Stations, BSs)必須負責傳送服務到它服務範圍之內所有的行動節點(Mobile Hosts, MHs)。BS 需要在同一時間內對許多 MH 提供 GPS、GIS、MC…等不同的服務，而其中和地理位置有關的查詢服務被稱為範圍查詢(Range Query, RQ)。RQ 的答案會依據當時查詢者的座標位置而改變，例如某一個 MH 在座標 A 及座標 B 發出相同“查詢離我 500 公尺之內的加油站資訊”的 RQ 時，它將會得到不一樣的答案。因此當 BS 為 MH 提供 RQ 服務時，需要知道 MH 現在的座標位置才能計算出正確的答案。然而，BS 所要服務的 MH 數量是很可觀的，如果大量的 MH 在短時間內同時發出 RQ 時，就會對 BS 造成極大的計算負擔。因為 BS 除了要更新 MH 目前的位置外，亦要計算各項 RQ 服務的答案，在有限的 BS 系統資源下，例如頻寬與計算能力，必然造成 BS 對某些 MH 的服務延遲，更甚至於造成服務中斷，以至於 BS 整體的服務效能下降。所以如何在 BS 系統資源的限制下，提供更快速的更新、查詢以及管理不同的 MH 位置與 RQ 服務，減輕 BS 的計算負擔，提升整體 BS 的服務效能，已經成為近年來一個重要的研究議題。

在以往的有線環境中，減輕伺服器端負擔及改善資料傳遞延遲的主要關鍵方法即為利用快取(caching)技術[4,3,15,9]。雖然可以將相同的方法應用在無線行動環境內以解決 BS 負擔過大的問題，但是在無線行動環境先天上的特性限制下，例如 MH 的儲存空間限制、頻寬限制、電池電力限制以及 MH 具備移動性(mobility)…等，直接應用先前的快取技術並不能完全可以適用於無線行動環境上，因此改善 BS 負擔的效果也將大打折扣。如今，新的無線通訊技術—無線隨意網路(Mobile Ad-hoc NETwork, MANET)的出現，例如 IEEE 802.11[5]，讓 MH 彼此之間可以不需要經由 BS 做連結，就能夠直接互相通訊，分享所需要的資料，也讓快取技術在行動環境上有了新的應用方向。

本文即是透過 MANET 的機制，提出行動節點協同快取(Mobile-host Cooperative Caching, MCC)方法，在 MH 發出新查詢之前，先搜尋在其 MANET 傳輸距離內的 MH，詢問是否有相似的查詢答案。如果鄰近 MH 有相似的查詢答案就可以重覆利用，如果鄰近 MH 沒有相似答案再向 BS 發出查詢請求。藉由 MCC 方法讓 MH 之間彼此分享先前所查詢過的 RQ 答案，如此 MH 便再也不用受到自己儲存空間的限制，可以重覆利用的資料大幅增加，因此就可以減少 MH 向 BS 發出的 RQ 數量。MCC 除了可以減輕 BS 的負擔之外，MH 與 BS 的通訊量也會減少，MH 下載資料的價錢花費亦跟著降低。

在 MCC 中，MH 有 5 種下載鄰近 MH 答案的方法，分別為依照 MH 傳輸距離遠近的先到先下載(First Come First Download, FCFD)、依照查詢答案範圍大小的最大範圍優先下載(Largest Range First Download, LRFD)、最小範圍優先下載(Smallest Range First Download, SRFD)、在所有查詢答案範圍找出範圍總合最大且不重疊的最大非重疊下載(Largest Non-Overlap Download, LNOD)以及 SRFD 及 LNOD 的混合方法 SRFD & LNOD Hybrid Download, SLHD)。另外，在本文中為了充分發揮 MH 的快取能力，能夠更有效率的減輕 BS 的計算負擔，亦針對 MH 的儲存空間做快取置換管理，讓 MH 可以儲存查詢頻率較高的答案，如此 MH 之間可以互相處理查詢的彈性將會更大，以達到提升 BS 服務效能目的。

最後，本文亦對各種下載方法做了不同的模擬實驗，實驗結果指出透過 MCC 方法將 BS 所要處理的查詢數分散給 MH 處理之後，可以大幅降低 BS 的負擔，有效提升 BS 的服務效能。而且導入快取管理的 MCC 效能表現更優於未導入快取管理的 MCC。

本文其餘章節組織如下：第二章為相關文獻探討；第三章描述 MCC 及不同下載方法的特性；第四章敘述模擬實驗環境設定並對所有下載方法做不同的實驗分析及比較；最後為結論及未來研究方向。

2. 相關文獻探討

在 RQ 服務中，BS 負擔來源主要有以下三類：

- **MH 位置更新的負擔**：MH 在移動時，BS 必須追蹤 MH 的座標位置，以便根據當時 MH 的位置和被查詢物件的位置做計算比對，才能提供正確的 RQ 答案。解決的方法可以利用空間索引方法。
- **RQ 答案更新的負擔**：當 MH 在移動時，有可能會去影響其他 MH 發出的 RQ 答案，例如車況查詢。因此 BS 就需要對被影響的 RQ 答案做更新。解決的方法可以記錄 RQ 的位置，當有查詢物件進入或離開 RQ 範圍時才做更新。
- **新 RQ 的負擔**：每當 MH 向 BS 發出新的查詢，相對於 BS 就是一個新的負擔，由於 BS 同時服務的 MH 數量極為龐大，因此新查詢所帶來的負擔是相當可觀的。解決的方法可以利用廣播方法將相同類型的 RQ 做一次處理，或者重覆利用先前所儲存的 RQ 答案，減少發出新 RQ 的次數。

根據 Myllymaki 和 Kaufman[6,7,8]的實驗結果顯示，查詢所帶來的代價(Cost)大約是更新代價的 3 倍，所以改善查詢所帶來的負擔，將可大幅提升 BS 的效能。而在改良查詢負擔的方法有 Multiple Query Processing (MQP)[12]，Validity Region[10]及 COoperative Caching (COCA)[2]。

MQP 在固定間隔時間內聚集所有的查詢，把具有共性(commonalities)的查詢找出，再將答案廣播給 MH。如圖 1(a)所示，Q1 和 Q5 擁有相同的答案，Q1 和 Q2 的答案有重疊的部分，Q3 的答案可以完全由 Q2 提供，Q4 則是完全獨立的部分。使用傳統查詢處理方式則如圖 1(a)，而使用 MQP 方式的查詢處理如圖 1(b)。因此 MQP 可以一次有效的處理大量的查詢並節省頻寬的浪費，但是如果廣播的資料太多，每筆資料等待被廣播的時間就會變得相當漫長，而且如果單位時間內查詢的共性太分散的話，MQP 的效能也會降低。

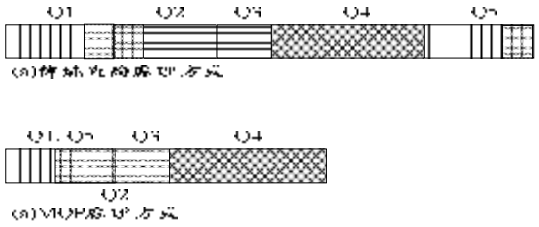

圖 1 MQP 範例

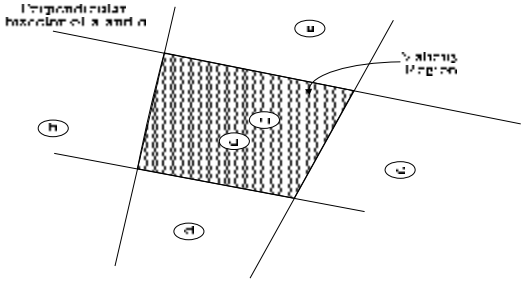

圖 2 Validity Region 範例

每當一個 MH 發出查詢後，BS 就會除了回傳查詢答案給 MH 之外，並計算出一個有效區(Validity Region)給 MH，只要 MH 還位於有效區內，代表 MH 上次查詢的答案仍然可以再次使用，MH 可以不用再發出新的查詢。但是只要 MH 一離開有效區，MH 就需要再發出新的查詢，BS

必須再重新計算 MH 的有效區。而有效區主要在於解決 RQ 內的最近物件(nearest neighbor)的問題，如圖 2 所示。q 為 MH 的位置，a、b、c、d、o 分別代表被查詢的物件，其中 o 為距離 q 最近的一個物件。只要依序在 a 與 o、b 與 o、c 與 o、d 與 o 之間找出垂直等分線，就可將其線段組合出有效區。但是，如果在 BS 內所要查詢的物件很密集，有效區就會變得很小，MH 離開 Validity Region 的機率就會變高。而計算出一個有效區平均需要求出 6 個邊(edge)[10]，因此，BS 的負擔就會與 MH 發出的查詢總數成線性成長。

不同於先前學者所提出的方法都只在於重複利用 MH 本身儲存的答案，COCA 內的 MH 可以互相分享鄰近 MH 儲存的答案以解決新查詢帶來的負擔。除了可以減少 BS 的負擔之外，即使 MH 不在 BS 的服務範圍內，MH 仍然可以嘗試從鄰近的 MH 下載答案。但是 COCA 並不是針對 PCS 環境的 RQ，因此 MH 所儲存的資料並不包含座標資訊，然而在 PCS 環境中，許多查詢都與 MH 及被查詢物件的位置有關。

3. MCC

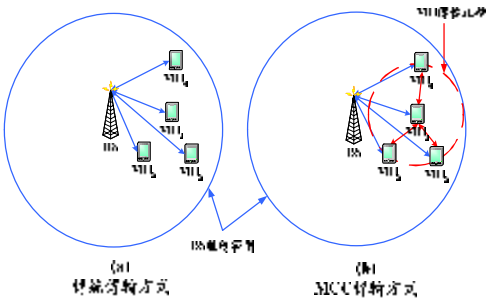

圖 3 傳統傳輸方式及 MCC 傳輸方式

在過去，RQ 查詢請求就如圖 3(a)一樣，MH 向 BS 發出查詢訊息，其中訊息資訊包含了 MH 現今的位置、欲查詢的物件及查詢的範圍…等，BS 收到 MH 的查詢請求後，將計算出的答案再回傳給 MH，也就是拉式(pull-based)資料傳遞模式。然而，當查詢數變多且集中在同一時間發出時，BS 有限的系統資源將成為整個 PCS 服務的瓶頸所在。如今，MANET 技術的出現，讓我們可以有效的解決這個問題。配備 MANET 的 MH，不需要連結 BS 即可與鄰近的 MH 連線組成一個動態群組，彼此能夠互相分享資料。因此，藉由 MANET 機制，我們提出了行動節點協同快取(Mobile-host Cooperative Caching, MCC)方法，讓 MH 間彼此分享快取過的查詢答案，以減少向 BS 發出的查詢數量。例如圖 3(b)，MH_1 在發出查詢前，會先詢問在其 MANET 傳輸距離內的 MH_2、MH_3 及 MH_4，只要在他們其中之一有相似或者相同的查詢答案，他們就會成為 MH_1 的候選 MH(Candidate MH, CMH)，MH_1 就可以選擇應該向那個 CMH 下載最適當的答案，以便滿足查詢要求。因此，藉由 MH 之間的互相幫助，MH_1 就可以縮小查詢範圍，甚至於可以不必向 BS 發出查詢。如此即可減輕 BS 所要處理的查詢負擔，又可以減少 MH 和 BS 之間的通訊量，避免額外的開銷。

3.1. 儲存架構

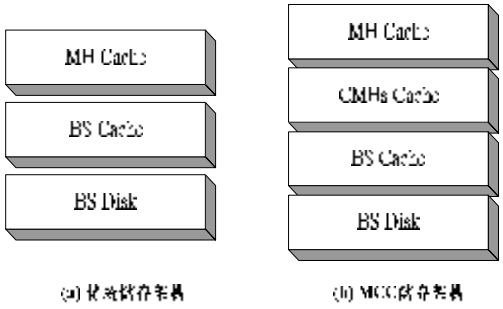

圖 4 儲存架構比較圖

在傳統行動環境中的查詢系統，其儲存架構階層分別包含：MH Cache、BS Cache 及 BS Disk，如圖 4(a)所示。當 MH 在發出查詢前，會先比對自己的快取儲存空間內是否有所需要的資料，如果無法在自己的快取儲存空間內找到所需資料時（也就是發生 cache miss），此時 MH 會轉向

BS 下載要查詢的資料。BS 在收到 MH 的查詢請求時，會先檢查 BS 的快取儲存空間是否存有 MH 所查詢的資料，如果 BS 無法在其快取儲存空間找到資料，BS 才會到磁碟空間尋找 MH 查詢的資料並傳給 MH。而 MCC 除了擁有三個傳統儲存架構階層之外，在 MH Cache 及 BS Cache 之間，額外新增了 CMHs Cache 儲存階層，如圖 4(b)所示。在 MCC 中，當 MH 本身的快取儲存空間內的資料無法提供查詢所要的答案時（此時為發生 local cache miss），MH 會先尋求 CMH 的快取儲存空間內是否有其所需要的資料，如果無法在 CMH 得到答案時或只得到部份答案時（此時為發生 global cache miss），MH 才會再向 BS 下載剩餘的答案，其餘的查詢過程就和一般傳統查詢一樣。

3.2. MH 架構

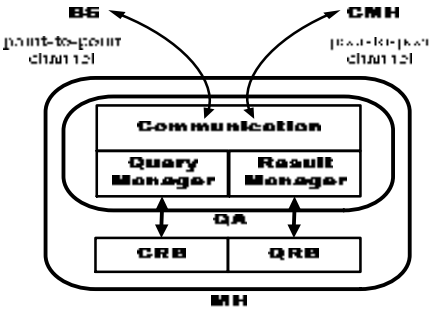

圖 5 MH 架構圖

圖 5 為 MCC 中的 MH 架構圖。MH 與 BS 之間藉由點對點傳輸頻道連結，MH 與 CMH 之間藉由對等式傳輸頻道連結。MH 由三個部份組成：Query Application (QA)與二個 Buffer，分別為 Candidate Result Buffer (CRB)以及 Query Result Buffer (QRB)。

QA 是整個 MH 架構的核心，由三個子元件組成，分別為 Communication、Query Manager 及 Result Manager，主要負責 MH 對外部的溝通及內部的管理。在對外的部份，Communication 就是負責與 BS 及 CMH 之間的連結，決定向 BS 或向 CMH 提出下載答案的請求訊息。對內的部份，Query Manager 管理 CRB，分析各個 CMH 可以提供的答案，以選擇向合適的 CMH 下載所需的答案。Result Manager 主要負責 QRB 的管理。當有其他 MH 要求分享答案時，QA 就會呼叫 Result Manager 到 QRB 找尋合適的答案。另外、當 QRB 已經沒有儲存空間時，如果再有新的答案要進來時，Result Manager 也必須負責判斷應該置換的位置，將最不重要的答案移出，讓新的答案可以被儲存進來。

CRB 主要負責儲存 CMH 的資訊。當 MH 發出廣播要求其他的 CMH 支援時，可能會收到一個以上的 CMH 的回應。此時這些 CMH 的資料，例如 CMH 的 ID 及可提供的答案，就會被儲存在 CRB 之內，QA 就可以根據 CRB 內的資料決定應該向那一個 CMH 下載適合的答案。

QRB 主要負責儲存答案的資訊。MCC 是同時利用其他 MH 快取過的答案，因此這些答案的範圍集合與查詢範圍相比之後，可能會有不足的部份。而這部份需要向 BS 下載後才能組合成為一個完整的答案。QRB 就是專門儲存這些完整答案的 Buffer。而 QA 也可以根據 QRB 內的儲存的資訊，決定應該提供一個適合的答案給其它的 MH 使用。

3.3. QR Protocol

在 MCC 中，對於 MH 之間分享查詢答案的過程，本文定義出一種協定，稱為 QR Protocol，讓 MH 之間可以很方便的透過 QR Protocol 進行協同快取合作。QR Protocol 包含二種協定，分別為 Query-Processing Protocol 與 Result-Responding Protocol。

在 Query-Processing Protocol 中，MH 有八種狀態，分述如下：

1. 檢閱(Inspect)：檢查 MH 本身的 QRB 內是否有所需的答案；
2. 請求(Request)：詢問鄰近 MH 是否有相似的答案；
3. CMH 選擇(Select-CMH)：分析 CRB 內的 CMH 資料，從中選擇適合的 CMH 以便下載所需的答案，選擇 CMH 的方法將在 3.4 節說明；
4. CMH 下載(Download-CMH)：從所選擇的 CMH 下載答案；
5. 比較(Compare)：比較查詢範圍和從其他 CMH 下載的答案總合是否有不足的部份；
6. BS 下載(Download-BS)：從 BS 下載答案；
7. 儲存(Store)：將答案儲存在 QRB 內。
8. 結束(End)：結束查詢。

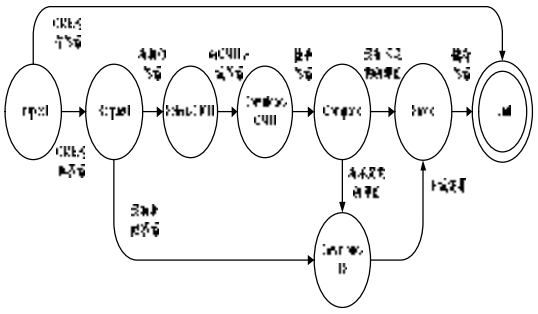

圖 6 Query- Processing Protocol 狀態轉換圖

圖 6 為 Query-Processing Protocol 狀態轉換圖。首先，MH 會先處於 Inspect 狀態，它檢查自己的 QRB 內是否已經有答案，如果已儲存答案，MH 本身就可以處理這次的查詢。如果沒有所需要的答案，MH 就轉為 Request 狀態時，它會先向鄰近的 MH 發出廣播，詢問它們是否有類似的答案，並且會設置等候時間，例如 10 秒鐘。如果在等候時間內，MH 沒有收到任何回應時，代表鄰近的 MH 都沒有相似的答案，MH 就進入到 Download-BS 狀態直接向 BS 請求答案，然後成為 Store 狀態。如果 MH 有收到鄰近 MH 的回應，它就將收到的訊息儲存在 CRB 內，接著轉成 Select-CMH 狀態，並運用第 3.4 節所提的方法從 CRB 內挑選要下載答案的 CMH，接著進入到 Download-CMH 狀態向選擇的 CMH 下載答案。從 CMH 下載完答案後，MH 就成為 Compare 狀態，它將會比對從 CMH 下載的答案總合是否和查詢有所誤差，如果有不足的部份就向 BS 下載，再儲存完整的答案到 QRB 內供其他 MH 使用，最後成為 End 狀態結束查詢。

Result-Responding Protocol 有五個狀態，分述如下：

1. 傾聽(Listen)：詢問鄰近 MH 是否有相似的答案；
2. 檢查(Check)：檢查 QRB 內是否有相似答案；
3. 回應(Reply)：回傳訊息，包含本身的 ID，位置，答案…等；
4. 傳輸(Transmit)：開始傳送答案；
5. 終結(Terminate)：結束回傳答案。

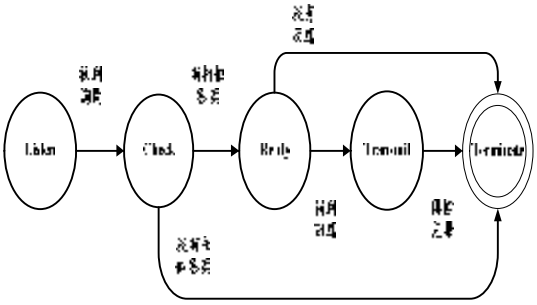

圖 7 Result-Responding Protocol 狀態轉換圖

圖 7 為 Result-Responding Protocol 狀態轉換圖。平常 MH 一直處於 Listen 狀態，一旦接受到其他 MH 的詢問廣播訊息時，MH 就會成為 Check 狀態。當 MH 進入到 Check 狀態時，會開始檢查 QRB 內是否有儲存相似的答案，如果沒有相似答案的話，MH 就跳到 Terminate 狀態。如果可以提供相似答案，MH 就進入到 Reply 狀態，這時 MH 就會回傳訊息給查詢者，並成為查詢者的 CMH。現在 CMH 會等待接收查詢者的回應，如果沒有

收到回應，代表其他的 CMH 可以提供更好的答案，因此直接跳到 Terminate 狀態。如果收到查詢者的下載請求，MH 就進入到 Transmit 狀態開始傳輸答案。當傳輸結束時，MH 就回到 Terminate 狀態。

3.4. CMH 的選擇

由於 CMH 回傳的答案可能有重覆，因此 MH 必須對 CRB 內的 CMH 做篩選，盡量避免下載到答案範圍重疊的部份。在本文中我們提出了五種不同的選擇方法，其中 R_Q 代表查詢範圍；CRB[i].ID 代表 CRB 內 CMH_i 的 ID，CRB[i].R 代表 CMH_i 的答案範圍，$1 \leqq i \leqq n$，n 為 MH 內 CRB 的個數。R 為 CRB[i].R 的總集合，初值為 $\varnothing$，被挑選的 CMH 的資訊則會暫存在 *AccessList* 中；圖 8 為答案下載範例，茲將各下載方法分述如下：

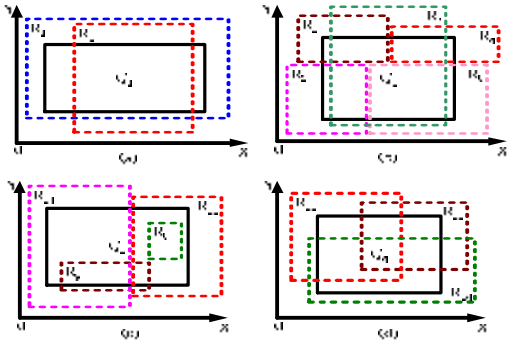

圖 8 下載範例

First Come First Download (FCFD)

FCFD 選擇 CMH 的方式是依照它們在 CRB 內的排列順序，當 MH 與 CMH 距離愈近時收到 CMH 回應的速度也愈快，也就是 FCFD 的選擇等同於 MH 與 CMH 距離遠近的順序。在 FCFD 中，首先 R 為空集合，接著 CRB[i].R 被依序與 R 做交集，如果 CRB[i].$R \not\subset R$，代表 CRB[i].R 可以提供原先 R 所沒有的部份，因此就將 R 設為 CRB[i].$R \cup R$，並將 CMH_i 的資訊存於 *AccessList* 中。此步驟一直持續到 R 可以完全包含 R_Q 的面積，或者瀏覽完整個 CRB 的資料為止。最後再檢查 *AccessList* 中是否有答案完全重覆 CMH_i，如果有提供重覆答案的 CMH_i，就將其移出 *AccessList*。

Largest Range First Download (LRFD)

LRFD 會先將 CRB 內的答案依範圍大小從大到小排列，再依序挑選 CRB[i].R 為下載對象。LRFD 的目的在於只要向少部份的 CMH 下載答案，其集合 R 即可滿足 R_Q，因為最大範圍的答案有可能一次就可以完全覆蓋全部的查詢範圍。如此 MH 和 CMH 之間的連結數及溝通所需的訊息量就會減少，傳遞資料所消耗的頻寬和電力也會較少。例如在圖 8(a)中，R_1 的面積可以全部覆蓋 Q_1 的面積，因此 LRFD 只需要下載 R_1 的答案而不需要考慮 R_2。但是優先選擇大範圍答案的下載時間較長，另外，答案之間重疊的機會也可能會變高，若 MH 下載到太多重覆的資料，除了浪費頻寬之外，更會消耗許多電池電力。例如圖 8(b)中，LRFD 會先從 R_7 開始下載，而在本範例中 LRFD 最終需要下載 R_3 到 R_7 才能滿足 Q_2，但 R_7 的資料將有許多部份成為重覆的資料。

Smallest Range First Download (SRFD)

SRFD 和 LRFD 相反，它是將 CRB 內的答案依範圍從小到大排列，再依其次序做挑選。SRFD 主要目的是避免 LRFD 可能下載到太多重覆資料的缺點。而且傳輸小範圍答案所需的時間也較短。再次考慮圖 8(b)的情形，SRFD 只需要下載 R_3 到 R_6 即可滿足 Q_2，而不會出現 LRFD 重覆下載的情形，因此可以省去無謂的頻寬浪費。但是因為 SRFD 是從小到大下載答案的特性，當 CMH 所提供的答案範圍太過於瑣碎時，CMH 的連結數就會比其他方法高出許多，下載到重覆資料的機率也會跟著變高。

Largest Non-Overlap Download (LNOD)

LNOD 是先從 CRB 內找出答案完全無重疊的組合，再從這些組合中找出範圍是最大的一組答案

集合。例如，在圖 8(c)中，不重疊的答案集合有{R_8, R_9}、{R_8, R_{11}}及{R_{10}, R_{11}}，其中{R_{10}, R_{11}}是這三組集合中涵蓋範圍最大的，因此{R_{10}, R_{11}}會成為 LNOD 的下載集合。因為 LNOD 只下載獨立的答案，所以它不會有 LRFD 的缺點，因此對 MH 的負擔是最小的。但是如果 CRB 內的答案皆互相重疊，LNOD 對減輕 BS 的負擔效果將變成最差的。例如圖 8(d)中，R_{12} 到 R_{14} 都互相有重疊的部份，因此 LNOD 只能挑選三者之中最大的一個下載，例如 R_{12}，其餘不足的部份只能再向 BS 發出請求來下載。因此 LNOD 是以 MH 的負擔為第一優先考量。

SRFD & LNOD Hybrid Download (SLHD)

最後一個方法是 SRFD 及 LNOD 的混合方法，SLHD 主要是融合 SRFD 及 LNOD 二個方法的優點。在下載其他答案前，MH 會先設定一個門檻值 *T*，接著先以 LNOD 選擇下載的答案，若 LNOD 所挑選的答案不能完全滿足查詢時，再將 CRB 內剩餘未被挑選的答案以 SRFD 的方法挑選，並計算重疊資料量是否其重疊部份的總合會超出 R_Q 面積 *T* 倍以上。如果重疊部份超出 *T* 倍以上，就不再挑選剩餘答案。SLHD 主要目的是以減輕 BS 的負擔為優先，如果要下載的答案集合超出 MH 的負擔，就改以減輕 MH 的負擔為優先，因此 *T* 值的設定會直接影響 BS 及 MH 雙方負擔的平衡。

3.5. MCC-DAU

快取技術是改善網路存取效能的重要關鍵技術，而快取置換機制更是影響整個快取效能的重要因素。先前小節中，MCC 的置換方法為最簡單的 FIFO。為了能夠更有效的提升 MCC 的效能，我們提出新的快取置換機制 MCC-DAU。考慮置換因素為範圍利用率(Domain)、存取率(Access)及更新率(Update)三種置換因素。

3.5.1. 置換代價函數

當 MH 內已無足夠的 QRB 儲存空間時，若再有新的資料要儲存，就必須將置換代價(replacement cost)最小的資料移除。而快取置換機制在作業系統、記憶體管理以及資料庫管理都曾被廣泛的被討論，這些快取置換機制的考慮通常包含二個因素，存取率：表示該筆資料在單位時間內被存取的頻率；更新率：表示該筆資料在單位時間內更新的頻率。本文在於利用 MH 儲存的範圍查詢答案以減輕 BS 負擔。因此在置換過程中，一筆資料的置換代價的考慮因素為範圍利用率、存取率以及更新率，說明如下：

範圍利用率 (Domain, *D*)

範圍利用率，以 *D* 表示，代表答案範圍與查詢範圍交集區域大小的程度。*D* 值愈高，代表交集區域愈大，該筆答案被利用的程度就會較高。範圍利用率的計算如公式(1)。

$$D_i = \alpha \times \frac{D_Q \cap D_{Ri}}{D_{Ri}} + (1 - \alpha) D_i^{old} \quad (1)$$

假設 α 為權重值，介於 0 至 1 之間；D_Q 為查詢範圍；D_{Ri} 為答案範圍；D_i^{old} 為上一次的範圍利用率，初值為 0。因此 $(D_Q \cap D_{Ri}) / D_{Ri}$ 表示 $D_Q \cap D_{Ri}$ 在 D_{Ri} 中所佔的比例。一筆答案的範圍利用率前後可能會有非常大的差異，因此需要利用權重值和前次的範圍利用率做運算，以逐漸平衡之間的變化。在公式中，D_i^{old} 佔的比重會比較重，主因在於範圍利用率是使用過去的歷史資料預測未來可能發生的範圍利用率。

存取率 (Access, *A*)

存取率，以 *A* 表示，代表該筆答案在單位時間內分享給其他 MH 的頻率。*A* 值愈高，代表該筆答案被其他 MH 存取的機率愈高。存取率的計算如公式(2)。

$$A_i = b \times \frac{1}{t_c - t_{la}} + (1 - b) \times A_i^{old} \quad (2)$$

假設 β 為權重值，介於 0 至 1 之間；t_c 與 t_{la} 分別為現在的時間與最近存取時間，單位為秒；A_i^{old} 為上次的存取率，初值為 0。$1/(t_c - t_{la})$ 表示存取時間間隔的倒數，當此值愈大時，代表 t_c-t_{la} 的值愈小，也就是距離上一次存取的時間愈近，因此也愈有可能在短期內再被存取。同樣地，每一次的存取率都會有不同的差異，因此也需要利用權重值和前次的存取率做運算，以平衡之間的變化。而存取率也是藉由過去的歷史資料去預測未來可能發生的存取，因此 A_i^{old} 所佔的比重會比較重。

更新率 (Update, *U*)

更新率，以 *U* 表示，代表答案更新頻率的程度。例如車流量查詢的更新率就很快，飯店位置查詢的更新率就是永不更新。*U* 值愈小，代表更新的時間間隔愈短，因此該筆答案資料的有效時間就很短。當一筆資料的期限超過有效期限就必須被淘汰。更新率的數學表示如公式(3)。*U* 值大於 0，且小於等於 1。當 *U* 值等於 1 時，代表該筆資料永不更新。

$$U_i = (0, 1] \quad (3)$$

範圍利用率、存取率及更新率三者的重要性為 *U*>*A*>*D*。如果一筆資料時常更新，時間有效性就非常差，即使先前的範圍利用率及存取率非常高，這筆資料也已經過期而無法被使用。而資料一旦被淘汰後，其範圍利用率及存取率就必須重新計算。所以更新率會同時影響範圍利用率及存取率。另外，一筆資料如果其範圍利用率非常高但存取率很低，代表該筆資料幾乎可以完全被利用但是卻很少被其他的 MH 存取。而減輕 BS 負擔主要是靠 MH 之間互相分享儲存的資料，因此存取率低的資料對於改善 BS 負擔較沒有幫助，所以必須先被淘汰。從以上對範圍利用率、存取率及更新率三者的重要性分析，可以推導出一筆資料 d_i 的置換代價函數如公式(4)。假設 μ 為權重值，介於 0 至 1 之間，而 A_i 所佔的比重會比 D_i 重。*cost* 值愈低的資料，代表置換此資料所造成的代價愈低，因此當有新資料要進入 QRB 時，要置換 *cost* 值最低的資料。

$$cost_i = (m D_i + (1 - m) A_i) \times U_i \quad (4)$$

3.5.2. MCC-DAU 範例說明

本小節將針對 MCC 快取方法提出範例說明，假設一個 MH 最多可同時儲存四筆答案資料，而四筆資料的各項參數資料列於表 1。

表 1 快取置換範例參數

參數 / 資料	$\frac{D_Q \cap D_{Ri}}{D_{Ri}}$	D_i^{old}	$\frac{1}{t_c - t_{la}}$	A_i^{old}	U_i
d_1	0.8	0.7	0.9	0.8	0.1
d_2	0.3	0.6	0.4	0.7	0.4
d_3	0.2	0.5	0.3	0.5	0.7
d_4	0.5	0.8	0.5	0.8	1

假設公式(1)的 α 為 0.3，公式(2)的 β 為 0.3，公式(4)的 μ 為 0.2，依據計算公式(1)可以算出 d_1 到 d_4 的 *D* 值，計算公式(2)可以算出 d_1 到 d_4 的 *A* 值，計算公式(4)可以算出 d_1 到 d_4 的 *cost* 值計算結果列於表 2。

表 2 快取置換範例計算結果

參數 / 資料	*D*	*A*	*U*	*cost*
D_1	0.73	0.83	0.1	0.081
D_2	0.51	0.61	0.4	0.236
D_3	0.41	0.44	0.7	0.3038
D_4	0.71	0.71	1	0.71

從表 2 可以看出 d_1 的 *D* 值和 *A* 值都高於 d_2、d_3 和 d_4，但因為 d_1 的 *U* 值太低，更新率太快，造成 d_1 很快就無效，因此 $cost_1$ 就遠低於其他的

cost，所以 d_1 是最先被置換的資料。同樣地，d2 和 d3 在 *D* 值和 *A* 值上的差異並不大，但 d_3 的更新率大於 d_2，所以置換 d_3 的代價高於置換 d_2。d_4 的 *D* 值、*A* 值及 *U* 值都相當高，是所有資料中最具有價值的，也因此 d_4 最應該被保留在 QRB 內。因此透過計算公式就可以很輕易的找出 *cost* 值最低的資料，並將之淘汰，所以可以有效地將最有價值的資料保留在 QRB 內。

4. 模擬實驗

在本章中，4.1 小節討論在 BS 的資料皆不更新的假設下，MCC 在 QRB 參數改變之下，QRB 置換為 FIFO，對改善 BS 查詢負擔的效能表現；4.2 節則比較在 BS 的資料會更新的假設下，MCC 在導入新的快取置換方法後的效能差異。我們以 JAVA 程式語言開發設計整個模擬實驗環境。模擬環境中，BS 的服務範圍為 $1000m^2$，MH 的個數 3000，隨意分散在 BS 服務範圍內。MH 的移動方向沒有任何限定，而且最高移動速度不能超過 V_{max}(27m/s)。每當 MH 要改變速度時，分別有 50%的機率變成低速、25%的機率變成中速、25%的機率變成高速；低、中、高速的範圍依序為 0 到 $\frac{Vmax}{3}$，$\frac{Vmax}{3}$ 到 $\frac{2Vmax}{3}$，以及 $\frac{2Vmax}{3}$ 到 V_{max}。MH 發出的查詢總數共 30000 個。每次查詢的範圍介於 $50m^2$ 到 $300m^2$ 之間。MH 之間最大的傳輸距離為 125m。QRB 值設定介於 1 到 20，代表 MH 每次最多只能儲存 1-20 組查詢答案。CRB 值設定為 50，表示若 MH 收到超過 50 個 CMH 的回應時，MH 最多只接受前 50 個 CMH 的訊息。每次 MH 要發出查詢時，必須等待 1 到 10 秒，避免查詢之間的間隔時間太短。另外在實驗中，我們假設每平方公尺的資料量大小為 1Kbit，而 MH 最大上傳速度為 200Kbps，查詢的種類總共有 10 類，例如地圖查詢和車況查詢即是二種不同的種類，更新率介於 0.25-1，查詢類型的更新速度則介於 15-30 秒。SLHD-T 代表使用 SLHD 方法，其門檻值 T 分別為 0.2 到 2，藉以觀察 SLHD 的變化。

4.1. MCC

快取是改良系統存取效能的關鍵技術，其中影響快取效能的因素之一為快取的儲存空間，因此在本小節的實驗中，將測試 QRB 數量的改變對 MCC 效能的影響。

圖為 QRB 個數與 BS 處理查詢次數關係圖，由圖中可以很明顯的看出隨著 QRB 個數的增加，BS 所要處理的查詢數大幅度的減少。在 QRB 的個數為 1 時，MH 只能儲存 1 個查詢答案，如果有新查詢答案要儲存時，舊答案馬上會被置換掉，因此 MH 之間可以分享的答案有限。但是由圖中可以看出即使 QRB 的個數只有 1，BS 所要處理的查詢數依然減至 60%-70%左右。而當 QRB 增加到 20 時，MH 幾乎可以將所有查詢過的答案儲存在 QRB 之內供其他的 MH 分享使用，因此 BS 處理的查詢數已經降到 23%-46%。由此可知，隨著 QRB 個數的增加，MCC 可以有效的將一半以上的查詢分擔給 MH 去做處理。在各種下載的方法中，FCFD、LRFD 及 SRFD 表現最好，且三種方法的曲線是一模一樣，原因在於這三種方法的主要差異在於下載 CMH 的順序，只要這些 CMH 內答案的集合可以滿足這一次的查詢，FCFD、LRFD 及 SRFD 都可以處理這個查詢。LNOD 在 QRB 大於 1 時，查詢數就會明顯比其他方法高出許多，差距範圍大約 5%-25%，原因在於 LNOD 的特性在於只挑選無重疊的答案集合，當 QRB 的個數增加時，MH 可儲存的答案也會增多，造成答案彼此重疊的機率也會增加，因此 LNOD 可選擇下載的答案就會變少，以至於查詢數比其他的方法高。SLHD 隨著 T 值的增加，可重覆下載的答案部份隨著增加，所以效能逐漸接近 FCFD、LRFD 及 SRFD，當 T 值等於 2 時，效能表現幾乎等於 SRFD。

圖 10 為 QRB 個數與 BS 處理查詢資料量關

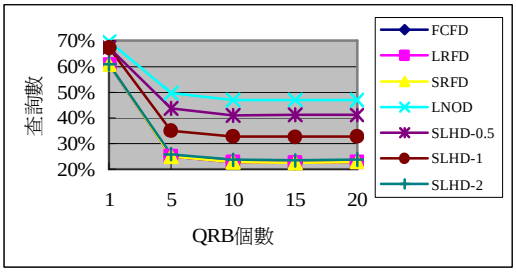

圖 9 QRB 個數與 BS 處理查詢次數

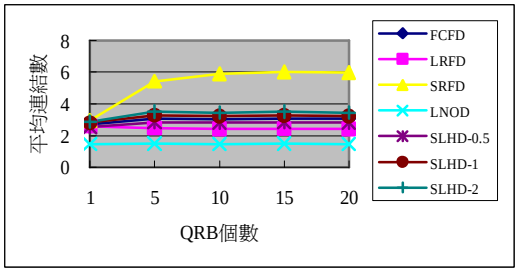

圖 11 QRB 個數與 MH 平均連結數

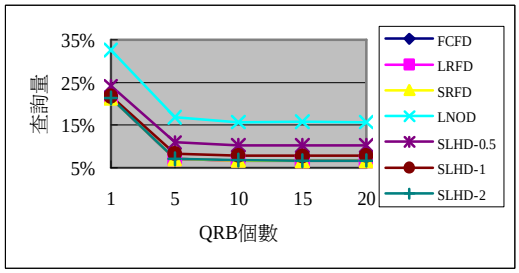

圖 10 QRB 個數與 BS 處理查詢資料量關係圖

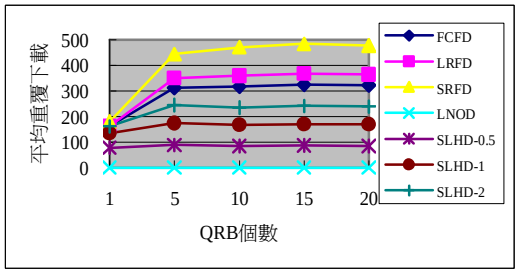

圖 12 QRB 個數與 MH 平均重覆下載資料量

係圖，也就是 BS 實際處理其他 MH 無法提供答案的部份。當 QRB 為 1 時，雖然 BS 仍然要處理 60%-70%左右的查詢數，但實際 BS 所要處理答案不足的部份只剩原先總處理查詢量的 21%-32%，這代表著就算 QRB 的值不高，只要靠 MH 互相幫助，這些查詢資料量依然有超過 70%以上可以由其他的 MH 做處理。而當 QRB 的個數大於 5 以上時，將近有 90%以上的資料量是由 MH 互相協力合作處理，所以 MCC 對於 BS 的查詢服務效能有非常大的改善。在所有的下載方法中，FCFD、LRFD 與 SRFD 所處理的總查詢數與其他方法相比最多約有 25%的差距，但是在處理的總查詢資料量中，各方法之間的差異已縮短到 3%-10%。因此只要 QRB 值夠大，LNOD 和 SLHD 的效能表現就會逼近 FCFD、LRFD 與 SRFD。當 QRB 大於 10 以上時，各方法的曲線亦趨於平緩，因此利用 MCC 改善 BS 查詢負擔，並不需要利用大量的 MH 儲存空間就能得到非常好的效果。

圖 11 為 QRB 個數與 MH 平均連結關係圖。MH 的平均連結數愈高，代表這筆答案必須由許多 CMH 提供不同的部份答案，MH 與 CMH 之間傳遞的訊息也就愈多，頻寬的浪費也會增加，因此連結數是愈少愈好。圖中很明顯的可以看出，當 QRB 大於 5 時，SRFD 的連結數比其他方法高出許多，這是因為每個 MH 之間可以儲存的答案個數增多，SRFD 就有可能會下載到許多小範圍的答案，以至於 SRFD 的連結數高於其他的方法。LNOD 擁有下載無重疊答案的特性，其連結數因此保持在 1 個到 2 個之間。LRFD 因為是先下載最大範圍的答案，所以可以很快將查詢範圍填滿，因此連結數僅次於 LNOD。SLHD 則是隨著 T 值的增加，可重覆下載的資料變大，下載的 CMH 個數也會漸漸增多，但仍然可以維持在 4 個連結數以內。由此可知，除了 SRFD 之外，MCC 可以把 MH 之間的連結數都不多，可以避免太多訊息傳遞造成頻寬的浪費。

圖 12 為 QRB 個數與 MH 平均重覆下載資料量關係圖。MH 之間可分享的答案可能會有重疊的部份，當下載的重疊部份愈多，所浪費的頻寬

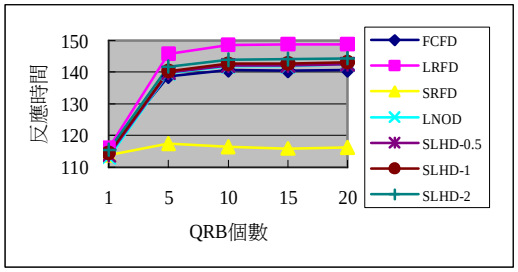

圖 13 QRB 個數與反應時間

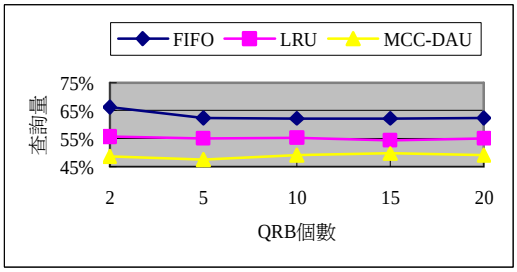

圖 15 置換方法與 BS 處理查詢資料量

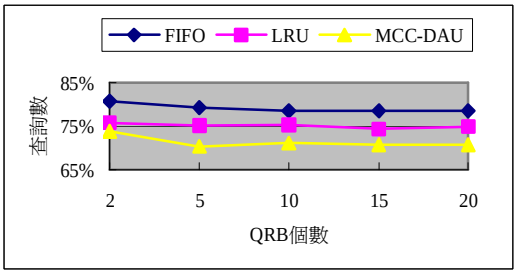

圖 14 置換方法與 BS 處理查詢次數

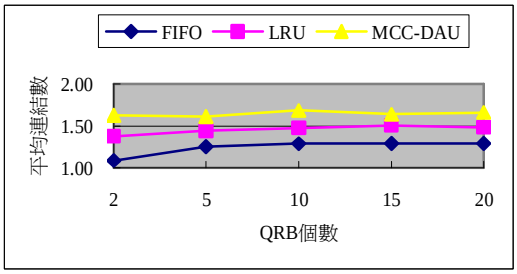

圖 16 置換方法與 MH 平均連結數

也愈多，這對 MH 是一種負擔，因此重覆下載的部份愈少愈好。當 QRB 值為 1 時，因為 MH 儲存的答案很容易就被置換掉，除了 LNOD 的重覆下載資料維持在 0 之外，各下載方法的重疊資料量都不大於 200。當 QRB 變大時，SRFD 因為其連結數較其他方法多，造成下載到重疊資料的部份也會比其他方法多，約為其他方法的 1.2 倍至 5 倍。LRFD 則因為優先下載大範圍的答案，當 CMH 之間能提供的答案範圍非常相似時，重疊的比例就會很高，所以重疊的資料量在所有方法中僅次於 SRFD。SLHD 隨著 T 值的減少，可重覆下載資料量就會減少，因此曲線會漸漸靠近 LNOD。QRB 變大時，MH 儲存大範圍答案的機率也會變高時，有可能只需要向少數 CMH 下載答案即可滿足查詢，所以連結數就不會隨著增加。而連結數維持在一定值時，答案之間重疊的機率也會變小，所以各方法的曲線走勢也會趨於平坦。

圖 13 為 QRB 個數與反應時間關係圖。CMH 可以提供的答案範圍愈大，MH 要下載的時間就愈長，MH 所消耗的電力就愈多，所以反應時間愈短愈好。在實驗中，假設 MH 可以同時向所有的 CMH 下載資料，因此反應時間的計算以最大範圍的答案為基準。由圖中可以明顯看出 LRFD 的反應時間是所有方法中最長的，而 SRFD 則是最短的，這是因為 LRFD 每次都先從大範圍的答案開始下載，而 SRFD 則是從小範圍的答案開始下載。LNOD 及 SLHD 都是有下載最大獨立答案集合的特性，因此效能表現都很接近。而 FCFD 的表現則介於 SRFD 及 LNOD 之間。

4.2. MCC-DAU

在本小節中，將比較 MCC-DAU 快取置換及其他快取置換方法的效能，包括：先進先出(First In First Out, FIFO)及最近最少使用(Least Recently Used, LRU)。另外，先前實驗已明白指出整體效能以 SLHD-1 的表現最佳，為了保持實驗圖的清晰度，以下實驗將只針對 SLHD-1 做探討。

圖 14 至圖 15 分別代表在不同的 QRB 個數，BS 處理查詢數及 BS 處理資料量關係圖。在圖 14

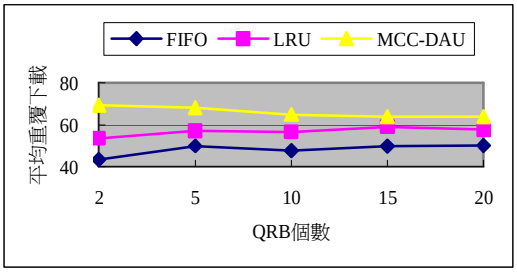

圖 17 置換方法與 MH 平均重覆下載資料量

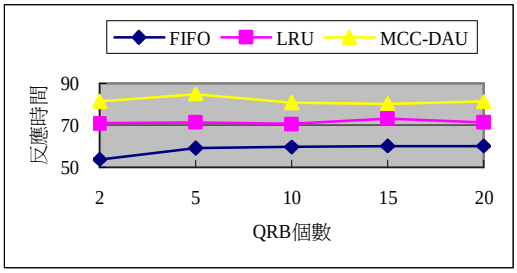

圖 18 置換方法與反應時間關係圖

至圖 15 中，MCC-DAU 所呈現的效能優於 FIFO 及 LRU。其主因在於 FIFO 的特性使得它每筆答案滯留在 QRB 內時間都不長，在 QRB 內的答案很快就會被置換出去，所以即使有筆答案可以常常被重覆利用，但留在 QRB 內的時間也不長，造成可被重覆利用的機會降少，以至於減輕 BS 負擔的效能表現為三個方法中最差的。LRU 則會將最近最少被使用的答案置換掉，然而要是有一筆答案的更新率非常高，但是卻在最近的時間內被使用了，於是 LRU 就會將該筆答案保留。但是這筆答案在 QRB 中卻很快的過期，而無法被其他的 MH 所利用，因此造成 LRU 效能變差。MCC-DAU 快取置換可以優先置換更新率高的答案，所以保留在 QRB 內的答案的時限性較高，並且可以保留利用率較高的答案，因此減輕查詢數及查詢量的表現最好。

圖 16 與圖 17 描繪 MH 平均連結數及 MH 平均重覆下載資料量關係圖。在三種置換方法中，FIFO 的連結數及平均重覆下載資料量都比其他二個方法少。理由如同前面所述，在 FIFO 置換方法下，每筆答案保存在 QRB 的時間都不長，很快就會被置換出去，進而減少該筆答案被重覆利用的機會，造成 MH 的連結數不高。而 MH 的連結數不高代表下載其他 MH 答案的機會減少，如此間接影響到重覆下載的資料量，所以 FIFO 連結數及重覆下載資料會比其他二個方法少。LRU 可以將常用的答案保留在 QRB 中，因此連結數會比 FIFO 多，所以重覆下載資料量也會比 FIFO 多。MCC-DAU 因為將存取率及利用率高的答案保留在 QRB 內，因此連結數及重覆下載資料為三個方法中最高。

圖 18 為反應時間關係圖。可以明顯看出 MCC-DAU 的反應時間最長，這是因為 MCC-DAU 可以將更新率低及利用率高的答案保留下來，因此在 QRB 內的答案範圍就會比其他方法大，所以下載的反應時間自然就比較長。而 LRU 比 FIFO 更能夠保留常用的答案，可重覆利用的答案自然會比 FIFO 多，因此 LRU 的反應時間也會比 FIFO 長。

5. 結論

本文針對 BS 在處理 RQ 查詢請求時，因為大量的 MH 同時發出查詢請求，快速消耗 BS 的系統資源，造成 BS 的服務效能不佳，而提出了行動節點協同快取(MCC)方法來解決這個問題。MCC 利用 MANET 機制，重覆利用鄰近 MH 先前儲存的相似答案，減少向 BS 發出重覆的查詢數量，降低 BS 的計算負擔。模擬實驗結果指出使用 MCC 方法後，在 BS 的資料不更新的假設下，BS 所要處理的查詢數降低約 50%以上。隨著查詢數變多，MH 可互相處理的查詢數也會跟著變多，BS 所要處理的查詢資料量也會大幅減少，因此 MCC 對於減輕 BS 的負擔有相當良好的表現。而在 BS 的資料會更新的假設下，MCC-DAU 可減少 BS30%左右的查詢數，而且改善的效能也比 FIFO 及 LRU 好。在所有的下載方法中，FCFD、SRFD 及 LRFD 改善 BS 負擔的效率。LNOD 改善 BS 負擔的效率雖然較差，但對 MH 所造成的負擔也較少。SLHD 表現介於中間，而 MH 及 BS 負擔的平均表現以 SLHD-1 為最佳。

本文以 MH 合作方式改良 BS 的查詢負擔，然而在答案的下載過程中會有重覆下載部份以至於增加 MH 的負擔。另外，對於超出 MH 傳輸距離之外的 MH 資料亦無法使用。因此在未來研究工作方面，可針對 CMH 答案重疊部份做切割，並設計一套良好的切割演算法。以及利用多點跳躍(multi-hop)的方法，讓 MH 可以透過 MH 做多點傳輸。

6. 參考文獻

1. Budiarto, N. Shojiro and T. Masahiko, "Data management issues in mobile and peer-to-peer environments", Data and Knowledge Engineering, (41)2, pp. 183-204, 2002.
2. C.Y. Chow, H.V. Leong and A. Chan, "Peer-to-Peer Cooperative Caching in a Hybrid Data Delivery Environment", Proceedings of the 2004 International Symposium on Parallel Architectures, Algorithms and Networks (ISPAN2004), 2004, pp.79-85.
3. C.Y. Chow, H.V. Leong and A. Chan, "Peer-to-Peer Cooperative Caching in Mobile Environments", Proceedings of the 24th International Conference on Distributed Computing Systems Workshops", pp. 528-533, 2004.
4. G. Cao, L. Yin and C.R. Das, "Cooperative Cache-Based Data Access in Ad Hoc Networks", IEEE Computer Magazine, 37(2), pp. 32-39, 2004.
5. IEEE-Std-802.11, "Wireless Lan Medium Access Control (MAC) and Physical Layer (PHY) Specification. 1999 Edition", 1999.
6. J. Myllymaki and J. Kaufman, "DynaMark: A benchmark for dynamic spatial indexing," Proceedings of the 4th International Conference on Mobile Data Management, 2003, pp.92-105.
7. J. Myllymaki and J. Kaufman, "High-performance spatial indexing for location-based services," Proceedings of the 12th international conference on World Wide Web, 2003, pp.112-117.
8. J. Myllymaki and J. Kaufman, "LOCUS - A Testbed for Dynamic Spatial Indexing," IEEE Data Engineering Bulletin (Special Issue on Indexing of Moving Objects), 2002, 25(2), pp.48-55.
9. J. Xu, Q. Hu, W.C. Lee and D.L. Lee, "Performance Evaluation of an Optimal Cache Replacement Policy for Wireless Data Dissemination", IEEE Transactions on Knowledge and Data Engineering, (16)1, pp. 125-139, 2004.
10. J. Zhang, M. Zhu, D. Papadias, Y. Tao, and D.L. Lee, "Location-based spatial queries," Proceedings of the 2003 ACM SIGMOD international conference on Management of data, 2003, pp.443-454.
11. O. Wolfson, B. Xu, S. Chamberlain and L. Jiang, "Moving Objects Databases: Issues and Solutions", Proceedings of the 10th International Conference on Scientific and Statistical Database Management, pp. 111-122, 1998.
12. R. Malladi and K.C. Davis, "Applying multiple query optimization in mobile databases," Proceedings of the 36th Annual Hawaii International Conference on System Sciences, 2003.
13. S. Dasbit and S. Mitra, "Challenges of computing in mobile cellular environment - a survey", Computer Communications, pp. 2090-2105, 2003.
14. S. Pierre, "Mobile computing and ubiquitous networking: concepts, technologies and challenges", Telematics and Informatics, (18)2,

pp. 109-131, 2001.

15. Y.L. Kwong, Z. Tari and P. Bertok, “Location-Aware Cache Replacement for Mobile Environments”, Proceedings of the IEEE Global Telecommunications Conference, pp. 3441-3447, 2004.